Remarks

The present Response is to the Office Action mailed 09/01/2009. Claims 1-28 are presented for examination.

Information Disclosure Statement

5. The information disclosure statement (IDS) submitted on 08/25/09 has been considered by the examiner. The information disclosure statement filed 08/25/09 fails to comply with 37 CFR 1.98(a)(2), which requires a legible copy of each cited foreign patent document; each nonpatent literature publication or that portion which caused it to be listed; and all other information or that portion which caused it to be listed. Please note the annotated information disclosure statement which highlights the references that are lacking an appropriately submitted copy.

Applicant's response:

Applicant is seeking to provide the references highlighted by the examiner, and will do so in a subsequent communication. As this does not interfere with a full response to the present action on the merits, the applicant is filing the rersponse. Applicant believes the IDS, as submitted, should comply with 37 CFR 1.98(a)(2).

Claim Rejections - 35 USC § 103

7. Claims 1-28 are rejected under 35 U.S.C. 103(a) as being unpatentable over DaCosta et al (US-6,826,553 11/30/04*)* in view of Weinberg et al (US-6,360,332 03/19/02) in further view of Heninger (US-6,029,207 02/22/00).

-In regard to substantially similar independent claims 1 and 12, DaCosta teaches an application for enabling automated notification of applied structural changes to electronic information pages on a network comprising:

an interface for enabling users to build and modify network navigation and interaction templates using functional logic blocks for automatically navigating to and interacting with interactive electronic information pages on the network (column 2, lines 11-30 & 55-67; column 3, lines 8-13, 35-43, & 53-65; column 5, lines 30-67; column 7, lines 17-54)(Figs. 1 & 7);

a navigation interface for integrating the software application to a proxy-navigation system for periodic execution of the templates (column 5, lines 19-20: "automatically repeat these steps in a scheduled manner or when requested");

a change notification module for indicating a navigation and interaction routine has failed and for creating a data file associated with the failed routine (column 18, lines 43-67: "it is known the script has failed...and proper notifications sent to individuals or entities responsible for the operation of the failing script by email...for example"; column 19, lines 1-15); and

sending proper notifications of the failed script to the developer upon failure of the script (column 6, lines 9-13 & 35-41; column 18, lines 53-67; column 19, lines 1-15). DaCosta does not specifically teach storing the data file in a data repository with a point-of-failure indication, parameters associated with the failed routine, and an identifier of the associated electronic information page subjected to the navigation. Weinberg teaches storing the data file (column 2, lines 39-40; column 6, lines 19-22), wherein the application periodically submits test navigation and interaction routines (column 6, lines 19-22), and upon failure of the routine, creates a data file (column 2, lines 39-40; column 3, lines 29-43; column 6, lines 19-22; column 17, lines 10-52)(Fig. 5F), the data file comprising a point-of-failure indication within the failed routine identifying the logic block of the template that failed (Fig. 5F: column 17, lines 17-21), parameters of the failure (column 17, lines 35-43), an identifier of the associated electronic page (columns 17-18: lines 62-12)(Fig. 5F: "URL: www.mercint.com"), and stores the data file in the data repository sending notification of the action to the developer (column 2, lines 39-40; column 6, lines 15-23). It would have been obvious to one of ordinary skill in the art at the time of the invention to have stored the failed navigation script of DaCosta and for the proper notifications of the failed script to have included a point in process of the failure along with the identifier of the associated web page, because Weinberg teaches that by storing the failed navigation script, a developer can easily display the results of the navigation and quickly determine the location of the failure of the routine (column 3, lines 29-44). This would have made the re-teaching (i.e. correcting) of the navigation script easier for the developer (column 6, lines 9-13 & 35-41; column 18, lines 42-67).

DaCosta teaches wherein functional logic blocks were part of the navigation and interaction templates containing all of the possible navigation and interaction instructions required by the navigation system-interface module as defined by the a given user/developer

(column 2, lines 20-31: "scripts...that locates and extracts data...precisely locating and extracting the select data with a granularity specified by the user" & lines 57-67: "capability for a user to specify...in an automated manor"; column 5, lines 39-55: "learn and store navigation paths...dialogs and forms that need to be filled...login name and password"; column 7, lines 16-28: "captures each user-generated event."; columns 7-8, lines 55-5: "automatically repeatedly query a web site...upon a single exemplematic query"; column 9, lines 5-44). Neither DaCosta nor Weinberg specifically teach wherein the defined functional logic blocks in the defined interaction scripts were modular parts of the interaction scripts. Heninger teaches building software components in a modular fashion such that each modular component could be constructed, modified, and tested independently (column 1, lines 20-29). It would have been obvious to one of ordinary skill in the art at the time of the invention for the functional logic blocks of DaCosta to have been modular parts of the navigation and interaction templates, because Heninger taught that computer software developers realize that modular interacting software components provide the advantages of being more easily designed, generated, tested, installed, and maintained as well leading to better computer products at a minimal cost (column 1, lines 20-67; column 2, lines 1-24). Thus the modular software components of Heninger would have provided the developers of DaCosta a better way of maintaining, editing, and correcting failed navigation scripts (column 18, lines 34-67) by allowing the developers to fix only the modular part of the failed navigation and interaction script.

Applicant's response

Applicant herein amends claims 1 and 12 to specifically recite that navigation templates are created by assembling a plurality of functional logic blocks and upon failure of a test routine, creates a data file, the data file comprising a point-of-failure indication within the failed routine identifying at least one functional logic block from the plurality of logic blocks in the associated template.

The Examiner states that DaCosta teaches; "an interface for enabling users to build and modify network navigation and interaction templates using functional logic blocks for automatically navigating to and interacting with interactive electronic information pages on the network (column 2, lines 11-30 & 55-67; column 3, lines 8-13, 35-43, & 53-65; column 5, lines 30-67; column 7, lines 17-54)(Figs. 1 & 7);" The Examiner further admits that, "Neither

DaCosta nor Weinberg specifically teach wherein the defined functional logic blocks in the defined interaction scripts were modular parts of the interaction scripts. Heninger teaches building software components in a modular fashion such that each modular component could be constructed, modified, and tested independently (column 1, lines 20-29).

Applicant points out that DaCosta/Heninger fail to teach assembling templates using a plurality of functional logic modules, as claimed. DaCosta specifically teaches that a navigation API includes a record module for recording a navigation path of a user for later playback. A data extraction API also includes a recording and playback module for selecting data to scrape. Specifically DaCosta teaches the data location and scraping tool of the invention comprises a browser plug-in to facilitate data collection, for example, scripts are added to the browser such as Microsoft Internet Explorer. Thus, the browser effectively serves as the operating system, and the scripts embedded in the browser form an input layer that locates and extracts data and effectively serves as a BIOS for retrieval of unstructured data.

Heninger column 1, lines 20-29 teaches a computer system, such as hardware being manufactured in modular form for independent testing, etc.. The portion of Heninger does continue to discuss software being written in modules, but specifically teaches that the modules must communicate with one another in order to create operating software, and thus are not autonomous modules (Col. 1, lines 45-51).

Applicant argues that one with skill in the art would not be capable of utilizing modular forms of operating software of Heninger with the navigation and extraction recordings of DaCosta in order to build navigation templates, as claimed. There is no motivation in DaCosta for breaking the script into modular portions because it is a single recording of a user's actions. Further, Heninger teaches an interdependence between software modules which would not accommodate identification of a single module during failure. One with skill in the art is aware that if a portion of intercommunicating modular software fails the entire application fails.

Additionally, applicant argues that the Examiner has not shown in the art the identification of a functional logic block in a failed navigation routine, as claimed. Applicant clearly claims that the software application notifies of failure instances of the executed routines, the failure instances include an identification of at least one of the logic blocks from the plurality of logic blocks in the associated template executed with the test and are logged in the database. In the Examiner's rejection it is stated that Weinberg teaches; " the data file comprising a point-

of-failure indication within the failed routine identifying the logic block of the template that failed (Fig. 5F: column 17, lines 17-21)". Said portion of Weinberg teaches:

"data in the form of a green check mark or a red "X" indicates whether the step passed or failed. In the example shown, an "X" is displayed next to the icon of iteration #4, indicating that at least one step in iteration #4 failed. By expanding the iteration #4 node, the user can view the details of the failure. Any verification step within a screen is provided one level indented from the screen."

Applicant argues that Weinberg also provides a facility for recording user interactions for input to a testing tool wherein the testing tool is specifically for testing the outputs of transactional servers. When a red X appears in the tree it is identifying an error output of the recorded input and identifies only a step where an erroneous output was given by the transaction server at the Web site. Weinberg clearly fails to identify a functional logic module used for automatic navigation and interaction within a Web site, as claimed (col. 16, line 7 to col. 17, line 9).

Applicant argues that one would not be motivated to utilize a transactional server output testing unit of Weinberg with navigation recording devices of DaCosta in order to accomplish identification of a failed functional logic block from a plurality of logic blocks used to build navigation and interaction scripts as claimed. Additionally, the art of Weinberg actually teaches executing recorded interactions within a stored Web page in order to successfully navigate the page. In applicant's invention, as claimed, modular functional logic blocks are used to build navigation templates, wherein when navigation fails only replacement or repair of a logic block occurs, sometimes automatically, rather than the requirement of re-recording user interactions and repairing transactional servers as in DaCosta and Weinberg. Applicant argues that the Examiner's reliance upon Weinberg's teaching of "upon future playback of said recorded series of steps, said system could record/indicate a point-in-process where a given step in the recorded sequence of steps failed." Does not read on actually indentifying a functional logic block from a plurality of logic blocks, as claimed. As previously argued, Weinberg marks a step as a failure when the transactional server returns an erroneous value, not when a logic block used for navigation and interaction fails.

Applicant believes claims 1 and 12 are patentable as amended and argued above. Claim 18 includes similar limitations and is also patentable as argued on behalf of claims 1 and 12. Claims 2-11, 13-17 and 19-28 are patentable on their own merits, or at least as depended from a patentable claim.

Summary

As all of the claims, as amended and argued above, have been shown to be patentable over the art presented by the Examiner, applicant respectfully requests reconsideration and the case be passed quickly to issue.

If any fees are due beyond fees paid with this amendment, authorization is made to deduct those fees from deposit account 50-0534. If any time extension is needed beyond any extension requested with this amendment, such extension is hereby requested.

Respectfully Submitted,
Tim Armandpour et al.

By */Donald R. Boys/*
Donald R. Boys
Reg. No. 35,074

Central Coast Patent Agency, Inc.
3 Hangar Way, Suite D
Watsonville, CA 95076
831-768-1755